



- (51) **International Patent Classification:**
G06F 17/00 (2006.01) *G06F 17/20* (2006.01)
- (21) **International Application Number:**
PCT/US2013/036183
- (22) **International Filing Date:**
11 April 2013 (11.04.2013)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (71) **Applicant:** HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P. [US/US]; 11445 Compaq Center Drive W., Houston, Texas 77070 (US).
- (72) **Inventors:** LEVI, Elad; Shabazi 19, 56100 Yehud (IL). MIZRAHI, Avigad; Shabazi 19, 56100 Yehud (IL). LUSTIG, Uriel; Shabazi 19, 56100 Yehud (IL).
- (74) **Agents:** SHOOKMAN, Jeb A. et al.; Hewlett-Packard Company, Intellectual Property Administration, 3404 E. Harmony Road, Mail Stop 35, Fort Collins, Colorado 80528 (US).
- (81) **Designated States** (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM,

DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

- (84) **Designated States** (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

- *as to the identity of the inventor (Rule 4.17(i))*
- *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

- *with international search report (Art. 21(3))*

(54) **Title:** AUTOMATED CONTEXTUAL-BASED SOFTWARE LOCALIZATION

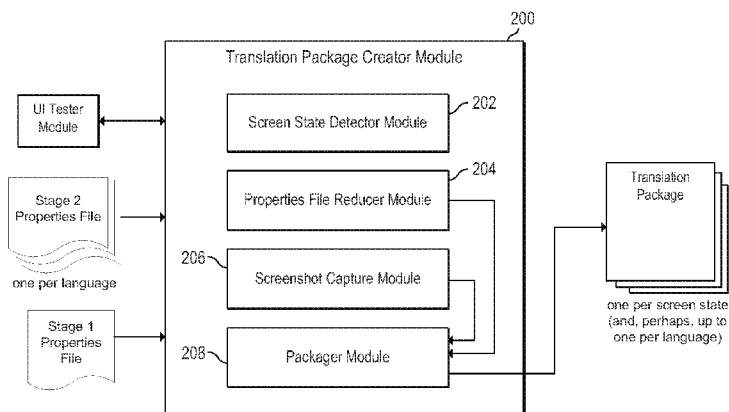


FIG. 2

(57) **Abstract:** Example embodiments relate to automated contextual-based software localization. In example embodiments, at least one stage computing device may automatically detect at least one screen state related to source code for a software product. The at least one stage computing device may automatically generate at least one reduced properties file, for a first language. Each of the reduced properties files may relate to one of the screen states. The at least one stage computing device may automatically create at least one screen shot. Each of the screen shots may relate to one of the screen states. The at least one stage computing device may automatically create at least one translation package, e.g., one for each screen state. Each translation package may include the screen shot and the reduced properties file associated with particular screen state.



AUTOMATED CONTEXTUAL-BASED SOFTWARE LOCALIZATION

BACKGROUND

[0001] In some scenarios, software products (e.g., applications, operating systems, etc.) may need to be provided in multiple different human languages. In some scenarios, a software product may be originally provided in a first language (e.g., a source language), and may need to be converted to at least one other language (e.g., target languages). For example, this conversion may be performed by identifying language-specific elements (e.g., displayable string elements within the user interface) of a software product and translating the string elements from the source language to the target language. Once all the language-specific elements have been translated the product may be re-compiled to produce a language-specific version of the software product for the target language. The process of converting a software product from one language to another language may be generally referred to as "localization." In some scenarios, a software product may be designed such that it can be adapted to various languages more easily, for example, by providing placeholders that may later be substituted with language-specific elements. In this scenario, the process of adapting such a software product may also be referred to as "localization."

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The following detailed description references the drawings, wherein:

[0003] FIG. 1A depicts block diagrams of stages and flows of information in an example software development process that utilizes automated contextual-based software localization;

[0004] FIG. 1B depicts block diagrams of stages and flows of information in an example software development process that utilizes automated contextual-based software localization;

[0005] FIG. 2 is a block diagram of an example translation package creator module for automated contextual-based software localization;

[0006] FIG. 3A depicts an example screen shot that may be included in an example translation package for automated contextual-based software localization;

[0007] FIG. 3B depicts at least part of an example reduced stage 1 properties file that may be included in an example translation package for automated contextual-based software localization;

[0008] FIG. 3C depicts at least part of an example reduced stage 1 properties file that may be included in an example translation package for automated contextual-based software localization;

[0009] FIG. 4A is a flowchart of an example method for automated contextual-based software localization;

[0010] FIG. 4B is a flowchart of an example method for automated contextual-based software localization;

[0011] FIG. 4C is a flowchart of an example method for automated contextual-based software localization;

[0012] FIG. 5 is a block diagram of an example stage computing device for automated contextual-based software localization; and

[0013] FIG. 6 is a flowchart of an example method for automated contextual-based software localization.

DETAILED DESCRIPTION

[0014] As described above, the localization process may include converting a software product from a first language (e.g., a source language) to at least one other language (e.g., target languages), or converting a software product designed with placeholders to at least one language. In some scenarios, the localization process may be used to convert a software product to accommodate not just different languages, but also different regions (e.g., slight regional preferences), different markets (e.g., different technical requirements) or other finer adjustments.

[0015] The localization process may be very time consuming and costly. The source code of the software product may need to be prepared, for example, by identifying and extracting all language-specific elements (e.g., displayable string elements within the user interface). Code analysis tools may be used, to some extent, to identify language specific elements, but developers are often not aware of these tools or forget about them. Then, when a list of extracted language-specific elements has been assembled, the language-specific elements need to be translated. While some automatic translation programs or services may be available, such services, in many scenarios, are not sophisticated enough to provide a high-quality translation. Typically, the translation of strings is performed by a human translator who is provided with a file or table of extracted strings that require translation. The human translator will then translate the strings and return the translations to the developer, who may insert the translations into the software product. Before inserting the translations into the software product, a linguistic reviewer (e.g., a natural speaker of the target language) may review the file or table of translated strings. Additional review of the translations may be performed once the translations are inserted and the software product is re-compiled and run. Such reviews are labor intensive and can be extremely expensive. Additionally, such reviews may take a long time, which may result in a slow delivery of localized versions of software products.

[0016] Linguistic errors are a major problem for localized software. Linguistic errors may refer to errors in the translation of the software product that are easily detected by native users of the software product. These errors may reflect poorly on the overall quality of the software product, and may even cause the software product to malfunction (e.g., crash). Fixing linguistic errors, especially at a later stage of the software product development, may require significant time and money (e.g., for translators and engineers to design and test fixes). One major cause of linguistic errors is that the human translation, and perhaps at least one round of review, are performed in isolation, e.g., without the precise context of how the strings are used in the software product. For example, in some languages, a correct translation of a

word or phrase for use on a button may be different from the correct translation for use in a title or in a text box. It may be very difficult if not impossible for a translator or reviewer to provide accurate translations when the strings are lacking context. Even if the translator or reviewer is familiar with the software product, it may be hard to picture the specific location and context of a string with the software product user interface.

[0017] The present disclosure describes automated contextual-based software localization. The present disclosure describes providing (e.g., to a human translator) language-specific elements that may need to be localized along with contextual information (e.g., screen shots) of how the elements are used in the software product user interface. The present disclosure describes automatically detecting multiple screen states in source code related to a software product, and automatically generating, for each of the multiple screen states, a screen shot and at least one reduced properties file that includes language specific elements displayable in such a screen shot. The present disclosure describes automatically creating, for each of the multiple screen states, a translation package that includes the screen shot and at least one reduced properties file associated with the particular screen state. The present disclosure may utilize automatic routines to extract language-specific elements from the source code. It should be understood that the techniques of the present disclosure may be used for source code that has been programmed for a particular source language and/or source code that has been programmed with placeholders for language-specific elements.

[0018] The present disclosure may provide benefits over previous localization routines. For example, human translations may be more accurate and may take less time because a translator may be able to refer to contextual information during the translation process. Human translators may need to ask far less or no clarifying questions (e.g., questions regarding whether a particular translation is correct) to an R&D team, which may save large amounts of time (e.g., one week per round of clarifying questions). Human translators may make far fewer linguistic errors during the translation process which may result in a more polished and more functionally correct

software product. Additionally, because of the streamlining of human translation in the software development process, the software development process may be much more scalable.

[0019] FIGS. 1A and 1B depict block diagrams of stages and flows of information in an example software development process that utilizes automated contextual-based software localization. For example, stages 102, 104, 106, 108 and 110 may be included in an example software development process, according to at least one embodiment of the present disclosure. Such a software development process may include more or less stages than are shown in FIGS. 1A and 1B. For example, the software development process may include additional stages for programming of the software, compiling the software, releasing the software, functionally testing the software (e.g., for bugs, performance, etc.) and the like. Any of these additional stages may be inserted between any of the stages depicted in FIGS. 1A and 1B and any of the stages depicted in FIGS. 1A and 1B may be inserted between, any stages in an existing software development process. In this respect, the stages depicted in FIGS. 1A and 1B may be integrated with any existing software development process (e.g., an Agile development process).

[0020] The term "software development process" (also referred to as "software development lifecycle" or "end-to-end (E2E) software development") as used herein may refer to any process for developing software that conforms to a structure, model or standard. Such a standard may define stages and tasks that may be required to develop, test and maintain a software product. Agile is one example standard that defines an iterative development process that relies on regular releases, regular testing and feedback. The term "stage" as used herein may be used to refer to a part of a software development process, for example, where at least one routine may be performed to progress the development of the software product. A stage may be carried out or executed on a computing device (e.g., referred to as a "stage computing device"). A stage computing device may execute a stage either automatically or with user input. Throughout this disclosure, it should be understood that the term "stage" may be

used in a flexible manner to refer to a stage of a software development process in the abstract or to a stage computing device used to perform the routines of a stage.

[0021] Each of stages 102, 104, 106, 108 and 110 may be performed by at least one computing device (e.g., referred to as a stage computing device), which may be, for example, any computing device capable of communicating (e.g., over a network) with at least one other computing device. In some embodiments, each stage may be implemented in a different computing device. In other embodiments, two or more of the stages shown in FIGS. 1A and 1B may be implemented in the same computing device. In other embodiments, at least one stage may be implemented by more than one computing device. The term "system" may be used to refer to either a single computing device or multiple computing devices, for example, where each computing device in the system is in communication with at least one other of the multiple computing devices in the system. More details regarding example computing devices that may be used for at least one of these stages may be described below, for example, with regard to stage computing device 500 of FIG. 5.

[0022] As a starting point to discussing stages 102, 104, 106, 108 and 110, source code for a software product may be available. The source code and perhaps other files may be used to build an executable version of the software product. The term "code base" may be used to refer to all the human-written files (e.g., as opposed to tool-generated files), including the source code, that are required to perform such as software build. Files in the code base other than the source code may include configuration files, properties files, resource files and the like. The source code or code base may be provided by a different stage (e.g., not shown in FIGS. 1A or 1B) of the software development process or as input by a user.

[0023] Stage 1 (indicated by reference number 102) may include analyzing the source code of a software product to identify language-specific elements and providing updated source code and at least one properties file. A stage computing device that implements stage 1 may receive (e.g., from an external computing device, user input or internal storage) stage 0 source code 112.

Stage 0 source code 112 may refer to the source code as it exists in whatever state it was in before stage 1. As one example, stage 0 source code 112 may have been programmed and functionally tested to at least some degree. A stage 1 computing device may include a properties extractor module 114 that may analyze stage 0 source code 112 and provide stage 1 source code 116 and a stage 1 properties file 118.

[0024] Properties extractor module 114 may analyze stage 0 source code 112 to identify language-specific elements. A language-specific element may refer to any text in the source code that may affect the way (e.g., in a language or location-specific manner) the related software product may display to a user (e.g., via a user interface). These language-specific elements may be the items that need to be “localized” in order to convert the software product to a different language. For example, strings in the source code that may be displayed to a user as part of a user interface (e.g., window, button, menu, toolbar, etc.) may be language specific elements. Other examples of language-specific elements include hotkeys, coordinate sizes or any other elements that may affect the way the software product is displayed depending on the target language or region. Properties extractor module 114 may detect code in the source code that indicates that a text string will be displayed to a user. Properties extractor module 114 may ignore programming comments, module names and the like, for example, because the precise text of these elements may not be displayed to a user. In some scenarios, properties extractor 114 may detect placeholders that were previously inserted into the source code with the intention of later being substituted with language-specific elements. Properties extractor may allow a user (e.g., a member of a software build team) to search (e.g., at least partially manually) for language-specific elements or it may perform the detection automatically.

[0025] Properties extractor module 114 may replace language-specific elements or placeholders in the source code with new placeholders, referred to herein as “property keys.” A property key may be a name or a text pattern that is unique to the language-specific element or placeholder that the

property key replaces. A property key may be similar to a constant used in programming, where the constant name may be inserted into the code and may take on particular designated value when the code is compiled. As explained below, later on, at least one language-specific value (e.g., translated value) may be associated with each property key. Properties extractor module 114 may step through stage 0 source code 112, and each time a language-specific element (e.g., a displayable string or a placeholder) is detected, properties extractor module 114 may generate a unique property key and replace the language-specific element with the property key. Properties extractor module may output stage 1 source code 116, which may be the same as stage 0 source code but with language-specific elements replaced with property keys.

[0026] Properties extractor module 114 may generate a properties file, for example, stage 1 properties file 118. Stage 1 properties file 118 may include a list of all the property keys generated by the properties extractor module 114 while it stepped through stage 0 source code 112. In some embodiments, properties extractor module 114 may generate more than one properties file, for example, where each properties file includes a subset of all the property keys generated. In fact, in various embodiments, any one of the properties files discussed herein may be implemented as more than one properties file. However, for simplicity, the present disclosure may refer to various single properties files. Stage 1 properties file 118 may also include a value for each property key. The property key values may be used, later on, when the source code is compiled, in which case, the values may replace the property keys in the source code. For the purposes of stage 1, the property key values may be initialized with the original values of the language-specific elements. For example, if a string was replaced with a property key, then the value associated with that property key in the properties file may be the string. As one specific example, if an original version of the software product was provided in English, then stage 1 properties file 118 may include property key values that are in English. As another example, if an original placeholder was replaced with a property key, then the value associated with that property key

in the properties file may be the original placeholder or the value may be empty.

[0027] Stage 1 may output stage 1 source code 116 and stage 1 properties file 118, and at least one later stage of the software development process may use these. For example, a stage 1 computing device may communicate stage 1 source code 116 and stage 1 properties file 118 to a stage 2 computing device. In some embodiments, a stage 1 computing device may automatically communicate stage 1 source code 116 and stage 1 properties file 118 to a subsequent stage (e.g., stage 2) as soon as the properties extractor 114 has generated the stage 1 source code 116 and/or stage 1 properties file 118. In some scenarios, the stage 1 computing device may communicate these items to at least one other stage, and then the at least one other stage may communicate these items to stage 2. In some scenarios, the stage 1 computing device may communicate these items to a stage that is later in the process than stage 2, for example, stage 3. In this respect, stage 2 may be skipped or excluded. In some scenarios, the software development process may include an additional stage (indicated by reference number 119) where the stage 1 source code 120 may be compiled based on the stage 2 properties file(s) 126.

[0028] Stage 2 (indicated by reference number 104) may include analyzing the stage 1 properties file, providing initial translations for at least one target language and generating at least one target language properties file. A stage computing device that implements stage 2 may receive (e.g., from a stage 1 computing device or from internal storage) stage 1 source code 120 and stage 1 properties file 122. Stage 1 source code 120 may be the same as or a copy of stage 1 source code 116, and stage 1 properties file 122 may be the same as or a copy of stage 1 properties file 118. Stage 2 may pass stage 1 source code 120 on to later stages, for example, without modification. A stage 2 computing device may include a translation service module 124 that may analyze stage 1 properties file 122 and provide at least one stage 2 properties file 126.

[0029] Translation service module 124 may analyze stage 1 properties file 122. Translation service module 124 may step through properties file 122, and for each property key, translation service module 124 may translate the associated property key value from the source language to at least one target language. Translation service module 124 may include or access a service that is capable of performing automatic translation of words and/or phrases. For example, translation service module 124 may include or have access to a translation repository. Additionally, translation service module 124 may include or have access to a translation provider (e.g., an online translation provider accessible via an API). As described above, while some automatic translation programs or services may be available, such services, in many scenarios, are not sophisticated enough to provide a high-quality translation. Therefore, stage 2 may be thought of as providing a "first draft" of translations for the stage 1 properties file. As indicated above, in some embodiments or scenarios, stage 2 may be skipped or excluded. In these situations, a first draft translation may not be provided for the properties file (e.g., stage 1 properties file 118) before stage 3. In these situations, a localized version of the software may not be available until later stages (e.g., stage 5).

[0030] Translation service module 124 may generate one stage 2 properties file 126 for each target language (e.g., for each language for which the original software product will be provided and/or supported). In this respect, the present disclosure may support localization to multiple target languages (i.e., "supported languages") simultaneously. Each stage 2 properties file 126 may include a list of all the property keys generated by the properties extractor module 114. Each stage 2 properties file 126 may also include a value for each property key. For the purposes of stage 2, the property key values may differ between the stage 2 properties files, depending on the target language. For example, if an original version of the software product was provided in English, then stage 1 properties file 122 may include property key values that are in English. Then, a first stage 2 properties file 126 may include property key values that are in French, and a

second stage 2 properties file 126 may include property key values that are in German, and so on.

[0031] Stage 2 may output stage 1 source code 120 (e.g., passed to FIG. 1B via circle "A"), stage 2 properties file(s) 126 (e.g., passed to FIG. 1B via circle "B") and stage 1 properties file 122 (e.g., passed to FIG. 1B via circle "C"), and a later stage may use these. For example, a stage 2 computing device may communicate these items to a stage 3 computing device. In some scenarios, the stage 2 computing device may communicate these items to at least one other stage, and then the at least one other stage may communicate these items to stage 3. In some scenarios, the stage 2 computing device may communicate these items to a stage that is later in the process than stage 3. In some scenarios, the software development process may include an additional stage (indicated by reference number 127) where the stage 1 source code 120 may be compiled based on the stage 2 properties file(s) 126.

[0032] Stage 3 (indicated by reference number 106) may include analyzing the source code and properties files to generate a translation package (TP) for each screen state (and perhaps up to one per language) of the software product. The term "screen state" may refer to a discrete part of a user interface (UI) of a software product that may be displayed to a user. A screen state may refer to the windows or layers that are displayed to a user, as opposed to the entirety of the screen that is presented to a user at any particular time. For example, if a main window were presented to a user, this may be a first screen state. Then, if a user clicked on a button and a smaller dialogue window popped up inside the main window, the smaller dialogue window may itself be a second screen state. In some embodiments, the UI information of a software product may be divided into screen states in a way such that no instances of language-based elements are duplicated between screen states. Thus, in the example from above, even while the smaller dialogue window is displaying, the first screen state would not include the same instances of language-based elements as the smaller dialogue window, and the second screen state would not include the same instances of language-based elements as the larger main window

behind the dialogue window. In this respect, each instance of a language-based element that is presented to a user throughout the usage of a software product may be uniquely associated with a particular screen state.

[0033] A stage computing device that implements stage 3 may receive (e.g., from a stage 2 computing device or from internal storage) stage 1 source code 128 (e.g., passed from FIG. 1A via circle "A"), stage 2 properties file(s) 130 (e.g., passed from FIG. 1A via circle "B") and stage 1 properties file 132 (e.g., passed from FIG. 1A via circle "C"). Stage 1 source code 128 may be the same as or a copy of stage 1 source code 120. Stage 2 properties file(s) 130 may be the same as or a copy of stage 2 properties file(s) 126, and stage 1 properties file 132 may be the same as or a copy of stage 1 properties file 122. A stage 3 computing device may include a UI tester module 134 and a translation package creator module 136. Each of these modules may include a series of instructions encoded on a machine-readable storage medium and executable by a processor of a stage computing device. In addition or as an alternative, each module may include one or more hardware devices including electronic circuitry for implementing the functionality described below. With respect to the modules described and shown herein, it should be understood that part or all of the executable instructions and/or electronic circuitry included within one module may, in alternate embodiments, be included in a different module shown in the figures or in a different module not shown.

[0034] UI tester module 134 may analyze and/or test the user interface (also referred to as the "graphical user interface" or GUI) components of stage 1 source code 128, for example, to test whether these components appear to function correctly with other parts of stage 1 source code 128. UI tester module 124 may scan or walk through the source code and identify code that may cause a UI component (e.g., a window, menu, etc.) to display to a user. An individual UI component may be considered a screen state. UI tester module 124 may simulate user interaction with the UI. For example, UI tester module 124 may simulate a user navigating through the graphical features of an application (e.g., clicking buttons, etc.). UI tester module 124 may ensure that every UI component (e.g., titles, navigation bars, buttons, etc.) that may

display to a user is analyzed. UI tester 124 may analyze and/or test the UI components of both device-based software products and web-based applications. UI tester module 124 may adapt to various code organization paradigms, for example, AOP (aspect oriented programming). In this respect, because translation package creator module 136 may interface with UI tester module 134, the localization techniques described herein may be used with any code organization paradigm.

[0035] Translation package creator module 136 may interface with UI tester module 134. In alternate embodiments, translation package creator module 136 may be integrated with UI tester module 134. Translation package creator 136 may use information from UI tester module 134 and stage 2 properties file(s) 130 to generate reduced properties files for each language and for each screen state (e.g., one for a main window, one for a smaller dialogue window, etc.). Each reduced properties file, for a particular language, may include only the language-specific elements (e.g., strings) that are presented to the user by the particular screen state. Thus, translation package creator 136 may use information from UI tester module 134 to determine which portions (e.g., property keys and values) of a stage 2 properties files 130 and/or stage 1 properties file are associated with each screen state. Translation package creator may also use information from UI tester module 134 to capture, receive and/or create a screen shot for each screen state.

[0036] FIG. 2 is a block diagram of an example translation package creator module 200, which may be similar to translation package creator module 136 of FIG. 1B, for example. Translation package creator module 200 may cooperate with a UI tester module (e.g., module 134) to automatically step through all the UI components of the source code to generate reduced properties files and a screen shot (e.g., used for software product context), for each screen state. Translation package creator module 200 may include a number of modules 202, 204, 206, 208. Each of these modules may include a series of instructions encoded on a machine-readable storage medium and executable by a processor of a stage computing device. In addition or as an

alternative, each module may include one or more hardware devices including electronic circuitry for implementing the functionality described below. With respect to the modules described and shown herein, it should be understood that part or all of the executable instructions and/or electronic circuitry included within one module may, in alternate embodiments, be included in a different module shown in the figures or in a different module not shown.

[0037] Screen stage detector module 202 may detect screen states in the source code (e.g., stage 1 source code 128). Screen stage detector module 202 may interface with a UI tester module (e.g., UI tester module 134) to detect screen states in the source code. Once a screen state is detected, translation package creator module 200 may (e.g., via module 202 and/or module 204) determine which portions of at least one properties file are associated with the screen state. For example, screen state detector module 202 may analyze the code associated with a particular screen state to determine which property keys are included in such code.

[0038] Properties file reducer module 204 may generate at least one reduced properties file based on at least one properties file and information about the screen state(s). For example, when the screen state detector module detects a screen state and the property keys associated with the screen state, properties file reducer module 204 may analyze each stage 2 properties file (e.g., one per language) to identify the same property keys (associated with the screen state). The portions of these properties files that are associated with the screen state may then be extracted and used to generate new properties files. The process may then be repeated for other screen states. The result may be, for each language-specific stage 2 properties file, one reduced properties file per screen state, where each reduced properties file includes property keys and values associated with the particular screen state. Properties file reducer module 204 may also generate a reduced properties file for the stage 1 properties file, in a similar manner.

[0039] Screen shot capture module 206 may capture, receive, generate and/or create at least one screen shot based on information about the screen state. For example, when the screen state detector module detects a screen

state, screen shot capture module 206 may capture, receive, generate and/or create an image (e.g., a GIF, JPG, PNG or the like) that is identical or similar to the image that a user sees when encountering the screen state while using the software product.

[0040] Packager module 208 may package, for each screen state, and perhaps up to one per language, the associated screen shot and at least one associated reduced properties file. Packager module 208 may receive (e.g., from module 206) a screen shot for each screen state. Packager module 208 may receive (e.g., from module 204) at least one reduced stage 2 properties file for each screen state. Packager module 208 may receive (e.g., from module 204) a reduced stage 1 properties file for each screen state. Packager module 208 may create a translation package (TP) for each screen state, and perhaps up to one language, as explained in more detail below.

[0041] Referring again to FIG. 1B, translation package creator module 136 may create at least one translation package (TP) 138, for example, one translation package per screen state, and perhaps, up to one per language. In various scenarios, translation packages may be constructed in various ways. In the specific embodiment of FIG. 1B, translation packages 138 may be constructed one per screen state, where each translation package 138 includes a screen shot 140, one reduced stage 2 properties file 142 for each supported language, and perhaps a reduced stage 1 properties file 144. In other scenarios, translation packages 138 may be constructed one per screen state, and then for each screen state, up to one translation package per supported language. In other words, in the scenario with one TP per supported language, the total number of translation packages may be [the number of screen states] times [the number of supported languages]. In this scenario, each translation package 138 (for a particular screen state and supported language) includes a screen shot 140, one reduced stage 2 properties file 142 for the particular supported language, and perhaps a reduced stage 1 properties file 144. In this scenario, for a particular screen shot, the translation packages (one per supported language) may include duplicate information. For example, each translation package (for a screen state) may include a copy of the associated

screen shot and, perhaps, a copy of the associated reduced stage 1 properties file. Just to be clear, in this scenario, translation packages 138 of FIG. 1B may appear with one screen shot, one reduced stage 1 properties file and one reduced stage 2 properties file, and then, the translation packages may be one per screen state and one per supported language. In other scenarios, some translation packages (per screen state) may include more or less reduced stage 2 properties files than other TPs. For example, one TP may support 3 languages and another TP may support 1 language.

[0042] Screen shot 140 may be an image (e.g., a GIF, JPG, PNG or the like) that shows a part of the UI that is displayed to the user in conjunction with the related screen state. Each reduced stage 2 properties file 142 may be a subset of the stage 2 properties file 130 for the corresponding language. Each reduced stage 2 properties file 142 may include only the portions of the state 2 properties file that are associated with the related screen state. For example, a stage 2 properties file may include only property keys and values that are associated with language-specific elements that may display as a result of the related screen state. Likewise, reduced stage 1 properties file 144 may be a subset of the stage 1 properties file 132 and may include only property keys and values from the stage 1 properties file 132 that are associated with language-specific elements that may display as a result of the related screen state.

[0043] Translation package creator module 136 may then package together screen shot 140, the at least one reduced stage 2 properties file 142 (e.g., one for each supported language), and perhaps a reduced stage 1 properties file 144. As explained above, in alternate embodiments, translation package creator module 136 may package one TP per screen state or one TP per screen state and up to one per supported language. Translation package 138 may take the form of any file or folder that may contain multiple files (e.g., a .zip file, a .rar file, a digital folder or the like). At this point, each TP may provide various elements that need to be localized (e.g., the property keys and values from the reduced stage 1 properties file 144), a first draft of translations for the elements (e.g., the values in the reduced stage 2 properties file(s), as well as application-specific contextual information for each element (the screen shot).

[0044] FIGS. 3A to 3C depict example contents of an example translation package. FIG. 3A depicts an example screen shot 300 that may be included in an example translation package. FIG. 3B depicts at least part of an example reduced stage 1 properties file 330 (e.g., in English) that may be included in an example translation package. FIG. 3C depicts at least part of an example reduced stage 2 properties file 360 (e.g., in French) that may be included in an example translation package. As can be seen by comparing FIGS. 3A, 3B and 3C, various language-specific elements of the UI (i.e., screen shot 300) have property key (and value) counterparts in the properties files. For example, language-specific elements (e.g., displayable strings) 302, 304, 306 and 308 in FIG. 3A have counterpart property keys/values 332, 334, 336 and 338 in FIG. 3B. Likewise, for example, language-specific elements (e.g., displayable strings) 302, 304, 306 and 308 in FIG. 3A have counterpart property keys/values 362, 364, 366 and 368 in FIG. 3C. As can be seen by comparing FIG. 3B and 3C, the property keys (e.g., "AddIntegrationDialogCancelButton") are the same in each properties file, but the property key values are different depending on the language (e.g., "Cancel" for English and "Annuler" for French).

[0045] FIGS. 3A to 3C show an example of how a translation package may provide context for various language-specific elements that need to be translated or localized. A translation package may provide a direct link between screens that are displayable in a software product and related language-specific elements displayed in such screens. A human translator, for example, may view the English properties file (e.g., FIG. 3B) to determine the values that need to be translated. The human translator may then view the "first draft" properties file for a particular language (e.g., French in FIG. 3C) to see a potentially correct translation for the values. The human translator may then view the screen shot (e.g., FIG. 3A) to gain context for how the values are used in the software product when displayed to a user. The human translator may then confirm that the first draft translation is correct, or may modify the translation to better fit the context.

[0046] Referring again to FIG. 1B, Stage 3 may output one translation package (TP) per screen state, and perhaps up to one per language, and a

later stage may use these. For example, a stage 3 computing device may communicate these items to at least one stage 4 computing device. In some scenarios, the stage 3 computing device may communicate these items to at least one other stage computing device, and then the at least one other stage computing device may communicate these items to at least one stage 4 computing device. In some scenarios, the stage 3 computing device may communicate these items to at least one stage computing device that is at a later stage in the process than stage 4. In some embodiments, a stage 3 computing device may automatically communicate translation packages 138 to a subsequent stage (e.g., stage 4) as soon as the translation package module 136 has generated the translation packages 138. Stage 3 may also output stage 1 source code 128 and/or stage 2 properties file(s) 130, and a later stage may use these. For example, a stage 3 computing device may communicate these items to a stage 5 computing device. In some scenarios, the stage 3 computing device may communicate these items to at least one other stage, and then the at least one other stage may communicate these items to stage 5. In some scenarios, the stage 3 computing device may communicate these items to a stage that is later in the process than stage 5.

[0047] Stage 4 (indicated by reference number 108) may include analyzing the translation packages to generate revised translation packages that may include more accurate translations than the "first draft" translations that may be included in translation packages from earlier stages. Stage 4 may include analysis and input by at least one human reviewer or translator (e.g., 148). Stage 4 may be implemented by more than one computing device, for example, up to one computing device per supported language. In this scenario, each computing device may receive input from a human translator of a different language. Stage 4 may receive (e.g., from a stage 3 computing device) translation packages 146. Translation packages 146 may be the same as or a copy of at least some of translation packages 138. As explained above, each translation package may include one reduced stage 2 properties file (e.g., for a particular screen state and a particular language), or may include more than one reduced stage 2 properties file. For example, if a human translator of a

particular stage 4 computing device can only translate to one target language, the translation packages may only include stage 2 properties files for that language. As another example, if a human translator can translate to all the supported language, the translation packages may include stage 2 properties files for all the supported languages. As explained above, translation packages may be automatically sent by a previous stage (e.g., stage 3). As a result, a human reviewer may receive an email notification or some other notification (e.g., generated and/or communicated by an earlier stage) that a translation package is ready to review or translate.

[0048] Human translator 148 may review translation packages 146. As explained above, the review and translation process for the human reviewer may be much easier because the translation packages include screen shots (to add contextual information) associated with reduced properties files, as well as first drafts of translations. Because the translation process has been made much easier, stage 4 may be implemented, for example, by online freelance translators (e.g., Amazon's Mechanical Turk or the like). Additionally, the translation times may be reduced which means later stages (e.g., stage 5) may begin sooner. Human translator 148 may create revised translation packages 150, for example, one per screen state (e.g., for the same screen states as in translation packages 146).

[0049] Each revised translation package 150 may include at least one revised reduced stage 2 properties files. For example, if the translation packages 146 each included multiple stage 2 properties files for different languages, then each revised translation package 150 may include multiple revised reduced stage 2 properties files. On the other hand, if the translation packages 146 each included a single stage 2 properties file (e.g., for a single target language), then each revised translation package 150 may include a single revised reduced stage 2 properties files. Each revised reduced stage 2 properties file 150 may be the same as a corresponding reduced stage 2 properties file in translation packages 146, but human translator 148 may have changed some of the property key values, for example, to provide a more accurate translation. Once the revised translation packages 150 are created,

they may be communicated or submitted to a later stage (e.g., stage 5) to be committed to the source code base. For example, a stage 4 computing device may include an interface by which human translator may indicate and submit revised translation packages, and then the revised translation packages may be automatically communicated to later stages (e.g., stage 5).

[0050] Thus, stage 4 may output one revised translation package per screen state, and perhaps, up to one per language, and a later stage may use these. For example, a stage 4 computing device may communicate these items to a stage 5 computing device. In some scenarios, the stage 4 computing device may communicate these items to at least one other stage, and then the at least one other stage may communicate these items to stage 5. In some scenarios, the stage 4 computing device may communicate these items to a stage that is later in the process than stage 5. In some scenarios, where multiple computing devices (e.g., for multiple human translators of different languages) are used for stage 4, each stage 4 computing device may output its associated translation packages, for example, one per screen state, for the language(s) that the computing device can handle.

[0051] Stage 5 (indicated by reference number 110) may include updating the code base based on the revised translation packages. A stage computing device that implements stage 5 may receive (e.g., from a stage 3 computing device) stage 1 source code 152 and stage 2 properties files 154. Stage 1 source code 152 may be the same as or a copy of stage 1 source code 128, and stage 2 properties files 154 may be the same as or a copy of stage 2 properties files 130. The stage 5 computing device may receive (e.g., from at least one stage 4 computing device) revised translation packages 156. Revised translation packages 156 may include the same or a copy of revised translation packages 150, and perhaps revised translation packages from other stage 4 computing device (e.g., that handle other languages). A stage 5 computing device may include a listener module (not shown) that detects when revised translation packages are submitted (e.g., by an interface in stage 4, by a human translator). The routines of stage 5 may begin automatically when revised translation packages are received. A stage 5

computing device may include a properties file updater module 158 which may generate at least one stage 5 properties file 160. This module may include a series of instructions encoded on a machine-readable storage medium and executable by a processor of a stage computing device. In addition or as an alternative, this module may include one or more hardware devices including electronic circuitry for implementing the functionality described below.

[0052] Properties file updater module 158 may analyze the stage 2 properties files 154 and the revised translation packages 156, and may update the values in the stage 2 properties files with any corresponding values in the revised translation packages 156. As one example method of updating, properties file updater module 158 may read a first stage 2 properties file 154 (e.g., for a first language). For each property key and value in the file, properties file updater module 158 may search the revised translation package 156 (e.g., only searching in revised reduced stage 2 properties files of the same language) for the same property key. If the same property key exists, properties file updater module 158 may replace (e.g., in memory) the property key value in the stage 2 properties file with the value from the revised reduced stage 2 properties file. Once all the property keys in the first stage 2 properties files have been searched for (and perhaps values replaced), properties file updater module 158 may generate a stage 5 properties file 160 for that language, for example, by writing the stage 2 properties file with replaced values to a new file. Properties file updater module 158 may repeat the above process for the rest of the stage 2 properties files (for other supported language), generating corresponding stage 5 properties files 160 for the other supported languages.

[0053] At this point, stage 5 may include an updated code base for the software product. The code base may include stage 1 source code 152 and stage 5 properties files (e.g., one per supported language). At this point, in this stage or a different stage (e.g., indicated by reference number 162), the source code may be compiled based on the stage 5 properties file. The source code may be compiled for every supported language. Thus, for example, multiple instances of the software product may be generated, one

for each supported language. Then, later, when a user is installing the software product, an installer may choose which instance of the application to install or provide based on various factors, for example, the user's country, region, locale or the like.

[0054] FIGS. 4A to 4C depict a flowchart of an example method 400 for automated contextual-based software localization. Method 400 may include multiple sub-methods, which, for simplicity, may simply be referred to as methods. For example, methods 430 and 460 may be part of method 400. Methods 400, 430, 460 may be described below as being executed or performed by a system, which may refer to either a single computing device (e.g., a stage computing device) or multiple computing devices, where these one or more computing devices may execute or perform at least one stage (e.g., stages 102, 104, 106, 108, 110) of a software development process. Methods 400, 430, 460 may be implemented in the form of executable instructions stored on at least one machine-readable storage medium, such as storage medium 520, and/or in the form of electronic circuitry. In alternate embodiments of the present disclosure, one or more steps of methods 400, 430, 460 may be executed substantially concurrently or in a different order than shown in FIGS. 4A to 4C. In alternate embodiments of the present disclosure, methods 400, 430, 460 may include more or less steps than are shown in FIGS. 4A to 4C. In some embodiments, one or more of the steps of methods 400, 430, 460 may, at certain times, be ongoing and/or may repeat.

[0055] FIG. 4A is a high-level flowchart of an example method 400 for automated contextual-based software localization. Method 400 may start at step 402 and continue to step 404, where a system (e.g., as part of stage 102) may analyze (e.g., via module 114) stage 0 source code (e.g., 112) to identify language-specific elements and provide updated source code (e.g., stage 1 source code 116) and a stage 1 properties file (e.g., 118). At step 406, the system (e.g., as part of stage 104) may analyze (e.g., via module 124) the stage 1 properties file (e.g., 122) to provide initial translations for at least one target language and generate at least one stage 2 properties file (e.g., 126) for the target language(s). At step 408, the system (e.g., as part of stage 106) may

execute or perform various routines as specified in more detail in method 430. At step 410, the system (e.g., as part of stage 108) may analyze (e.g., via human translator 148) translation packages (e.g., generated at step 408) to generate revised translation packages (e.g., 150) that may include more accurate translations. At step 412, the system (e.g., as part of stage 110) may execute or perform various routines as specified in more detail in method 460. Method 400 may eventually continue to step 450, where method 400 may stop.

[0056] FIG. 4B is a flowchart of an example method 430 for automated contextual-based software localization. Method 430 may be part of method 400, for example, substituted for step 408. Method 430 may be executed or performed as part of stage 106 in FIG. 1B, for example. Method 430 may start at step 432 and continue to step 434, where a system may detect (e.g., via module 134, 136 and/or 202) a screen state in stage 1 source code (e.g., 128). At this point, the steps included in box 436 (e.g., steps 438 and 440) may be executed once for each supported language. At step 438, the system may analyze (e.g., via module 136, 202 and/or 204) a stage 2 properties file (e.g., 130) for a particular language to determine portions that relate to the screen state. At step 440, the system may generate (e.g., via module 136 and/or 204) a reduced stage 2 properties file (e.g., 142) for the particular language, including only portions of the stage 2 properties file that relate to the screen state.

[0057] At step 442, the system may analyze (e.g., via module 136, 202 and/or 204) the stage 1 properties file (e.g., 132) to determine portions that relate to the screen state. At step 444, the system may generate (e.g., via module 136 and/or 204) a reduced stage 1 properties file (e.g., 144) that includes only portions of the stage 1 properties file that relate to screen state. At step 446, the system may capture (e.g., via module 134, 136, 202 and/or 206) a screen shot for the screen state. At step 448, the system may package (e.g., via module 136 and/or 208) the screen shot, the reduced stage 2 properties files and the reduced stage 1 properties file into one or more translation packages (e.g., 138), as described in more detail above. Method 430 may

then return to step 434 and repeat the method for each screen state detected in the source code.

[0058] FIG. 4C is a flowchart of an example method 460 for automated contextual-based software localization. Method 460 may be part of method 400, for example, substituted for step 412. Method 460 may be executed or performed as part of stage 110 in FIG. 1B, for example. Method 460 may start at step 462 and continue to the steps in box 464. The steps included in box 464 (e.g., steps in box 466 and step 474) may be executed once for each supported language. The method may continue to the steps included in box 466, which may be executed once for each property key included in the stage 2 properties file (e.g., 154) for the particular language. At step 468, a system may find (e.g., module 158) the first or next property key in the stage 2 properties file. At step 470, the system may search the revised reduced stage 2 properties files (e.g., in at least one revised translation package 156) for the particular language (and for all screen states) to find the same property key. At step 472, the system may replace (e.g., via module 158) the property key value in the stage 2 properties file with the value from the revised reduced stage 2 properties file in which the same property key was found. At step 474, the system may generate a stage 5 properties file (for the particular language) using the update property key values determined in the steps of box 466. Method 460 may eventually continue to step 476, where method 460 may stop.

[0059] FIG. 5 is a block diagram of an example stage computing device 500 for automated contextual-based software localization. Stage computing device 500 may be any computing device capable of communicating (e.g., over a network) with at least one other computing device. More details regarding stages that may utilize at least one stage computing device similar to stage computing device 500 may be described above, for example, with respect to FIGS 1A, 1B and 2. In the embodiment of FIG. 5, stage computing device 500 includes a processor 510 and a machine-readable storage medium 520.

[0060] Processor 510 may be one or more central processing units (CPUs), microprocessors, and/or other hardware devices suitable for retrieval and execution of instructions stored in machine-readable storage medium 520. In

the particular embodiment shown in FIG. 5, processor 510 may fetch, decode, and execute instructions 522, 524, 526, 528 to perform automated contextual-based software localization. As an alternative or in addition to retrieving and executing instructions, processor 510 may include one or more electronic circuits comprising a number of electronic components for performing the functionality of one or more of instructions in machine-readable storage medium 520 (e.g., instructions 522, 524, 526, 528). With respect to the executable instruction representations (e.g., boxes) described and shown herein, it should be understood that part or all of the executable instructions and/or electronic circuits included within one box may, in alternate embodiments, be included in a different box shown in the figures or in a different box not shown.

[0061] Machine-readable storage medium 520 may be any electronic, magnetic, optical, or other physical storage device that stores executable instructions. Thus, machine-readable storage medium 520 may be, for example, Random Access Memory (RAM), an Electrically-Erasable Programmable Read-Only Memory (EEPROM), a storage drive, an optical disc, and the like. Machine-readable storage medium 520 may be disposed within stage computing device 500, as shown in FIG. 5. In this situation, the executable instructions may be "installed" on the device 500. Alternatively, machine-readable storage medium 520 may be a portable (e.g., external) storage medium, for example, that allows stage computing device 500 to remotely execute the instructions or download the instructions from the storage medium. In this situation, the executable instructions may be part of an "installation package". As described herein, machine-readable storage medium 520 may be encoded with executable instructions for automated contextual-based software localization. Although the particular embodiment of FIG. 5 includes instructions that may be included in a particular stage (e.g., stage 106) of a software development process, computing device 500 may instead or in addition include instructions related to other stages of such a software development process.

[0062] Screen state detector instructions 522 may detect screen states in the source code (e.g., stage 1 source code 128) and may interface with a UI tester

module (e.g., UI tester module 134). Screen state detector instructions 522 may be similar to screen stage detector module 202 of FIG. 2, for example, and more details may be provided above with respect to screen stage detector module 202 and/or translation package creator module 136 of FIG. 1B. Properties file reducer instructions 524 may generate at least one reduced properties file based on at least one properties file and information about the screen state(s). Properties file reducer instructions 524 may be similar to properties file reducer module 204 of FIG. 2, for example, and more details may be provided above with respect to properties file reducer module 204 and/or translation package creator module 136 of FIG. 1B.

[0063] Screen shot capture instructions 526 may capture, receive, generate and/or create at least one screen shot based on information about the screen state. Screen shot capture instructions 526 may be similar to Screen shot capture module 206 of FIG. 2, for example, and more details may be provided above with respect to screen shot capture instructions 206 and/or translation package creator module 136 of FIG. 1B. Packager instructions 528 may package, for each screen state, and perhaps up to one per supported language, the associated screen shot and at least one associated reduced properties file. Packager instructions 528 may be similar to packager module 208 of FIG. 2, for example, and more details may be provided above with respect to packager module 208 and/or translation package creator module 136 of FIG. 1B.

[0064] FIG. 6 is a flowchart of an example method 600 for automated contextual-based software localization. Methods 600 may be described below as being executed or performed by a system, which may refer to either a single computing device (e.g., a stage computing device) or multiple computing devices, where these one or more computing devices may execute or perform at least one stage (e.g., stages 102, 104, 106, 108, 110) of a software development process. Method 600 may be implemented in the form of executable instructions stored on a machine-readable storage medium, such as storage medium 520, and/or in the form of electronic circuitry. In alternate embodiments of the present disclosure, one or more steps of method 600 may be executed substantially concurrently or in a different order than shown in FIG.

6. In alternate embodiments of the present disclosure, method 600 may include more or less steps than are shown in FIG. 6. In some embodiments, one or more of the steps of method 600 may, at certain times, be ongoing and/or may repeat.

[0065] Method 600 may start at step 602 and continue to step 604, where a system may detect (e.g., via instructions 522) at least one screen state in source code (e.g., stage 1 source code 128), for example, by using a UI tester module (e.g., UI tester module 134). At step 606, the system may generate (e.g., via instructions 524) a reduced properties file based on at least one properties file and information about the screen state(s). At step 608, the system may capture, receive, generate and/or create (e.g., via instructions 526) at least one screen shot based on information about the screen state. At step 610, the system may package (e.g., via instructions 528), for each screen state, and perhaps up to one per supported language, the associated screen shot and at least one associated reduced properties file. Method 600 may eventually continue to step 612, where method 600 may stop.

CLAIMS

1. A system for automated contextual-based software localization, the system comprising:

at least one processor to:

automatically detect multiple screen states in source code related to a software product, wherein each of multiple language-specific elements in the source code has been replaced with one unique property key of multiple property keys;

automatically generate, for each of the multiple screen states, a reduced properties file that includes a subset of the multiple property keys, wherein the property keys in the subset are uniquely related to the particular screen state, and wherein the reduced properties file further includes, for each included property key, a property key value related to a first language;

automatically generate, for each of the multiple screen states, a screen shot that is uniquely related to the particular screen state; and

automatically create, for each of the multiple screen states, a translation package that includes the screen shot and the reduced properties file associated with the particular screen state.

2. The system of claim 1, wherein the at least one processor is further to automatically send, upon creation, the translation packages respectively associated with the multiple screen states to a translation computing device that revises the property key values in the reduced properties files and creates, for each of the multiple screen states, a revised reduced properties file.

3. The system of claim 2, wherein the translation computing device receives input from a human translator to review the property key values.

4. The system of claim 2, wherein the at least one processor is further to:

receive from the translation computing device the revised reduced properties files; and

create a single properties file, for the first language, based on the revised reduced properties files, wherein the single properties file and the source code can be compiled to create an executable version of the software product.

5. The system of claim 1, wherein the at least one processor is further to:
automatically generate, for each of the multiple screen states, a second reduced properties file that includes the subset of the multiple property keys that is uniquely related to the particular screen state, wherein the second reduced properties file further includes, for each included property key, a property key value related to a second language; and

automatically create, for each of the multiple screen states, a second translation package that includes the screen shot and the second reduced properties file associated with the particular screen state.

6. The system of claim 1, wherein the at least one processor is further to:
automatically generate, for each of the multiple screen states, a second reduced properties file that includes the subset of the multiple property keys that is uniquely related to the particular screen state, wherein the second reduced properties file further includes, for each included property key, a property key value related to a second language,

wherein the translation package, for each of the multiple screen states, includes the second reduced properties file associated with the particular screen state.

7. The system of claim 1, wherein the at least one processor is further to:
run a user interface tester module that analyzes the user interface components in the source code,

wherein the user interface tester module is used to walk through the source code to automatically detect the multiple screen states and

automatically generate, for each of the multiple screen states, the reduced properties file.

8. A method for automated contextual-based software localization, the method comprising:

receiving, in association with a software product, source code and a first-language properties file, wherein language-specific elements in the source code have each been replaced with a unique property key of multiple property keys, and wherein the first-language properties file includes the unique property keys and associated first-language property key values, each first-language property key value being equal to the a first-language translation of one of the language-specific elements that the associated unique property key replaced;

automatically detecting a first screen state associated with the source code;

automatically generating a first reduced first-language properties file that includes a subset of information in the first-language properties file, wherein the subset uniquely relates to the first screen state;

automatically generating a first screen shot for the first screen state; and

automatically packaging, into a first translation package, the screen shot and the first reduced first-language properties file.

9. The method of claim 8, further comprising:

receiving, in association with the software product, an original-language properties file that includes the multiple unique property keys and associated original property key values, each original property key value being equal to one of the language-specific elements that the associated unique property key replaced;

automatically generating a first reduced original-language properties file that includes a subset of information in the original-language properties file, wherein the subset uniquely relates to the first screen state,

wherein the automatic packaging of the first translation package includes automatically packaging the first reduced original-language properties file into the first translation package.

10. The method of claim 8, further comprising automatically sending the first translation package to a translation service that reviews or modifies, via human translator input, the reduced first-language properties files in the first translation package to create a first revised reduced properties file.

11. The method of claim 10, further comprising:

receiving, from the translation service, the first revised reduced properties file related to the first screen state;

receiving, from the translation service, a second revised reduced properties file related to a second screen state; and

create a single revised first-language properties file based on the first revised reduced properties file and the second revised reduced properties file, wherein the single revised first-language properties file and the source code can be compiled to create an executable version of the software product.

12. The method of claim 10, further comprising:

receiving, from the translation service, the first revised reduced properties file related to the first screen state; and

compiling the source code using the first revised reduced properties file.

13. A machine-readable storage medium encoded with instructions executable by at least one processor of at least one stage computing device for automated contextual-based software localization, the machine-readable storage medium comprising:

instructions for automatically detecting multiple screen states related to source code for a software product;

instructions for automatically generating multiple reduced properties files, for a first language, wherein each of the multiple reduced properties files

relates to one of the multiple screen states and wherein the reduced properties files provide localization information that may be used to compile the source code for the first language;

instructions for automatically creating multiple screen shots, wherein each of the multiple screen shots relates to one of the multiple screen states; and

instructions for automatically creating multiple translation packages, one for each screen state, wherein each translation package includes the screen shot and the reduced properties file associated with particular screen state.

14. The machine-readable storage medium of claim 12, wherein the source code includes multiple language-specific elements that need to be localized to compile the source code for the first language, and wherein each of the multiple language-specific elements is uniquely associated with one of the multiple screen states.

15. The machine-readable storage medium of claim 13, wherein each of the reduced properties files associated with a particular screen state includes localized or translated versions of the language-specific elements that are uniquely associated with the particular screen state.

16. The machine-readable storage medium of claim 12, further comprising instructions for automatically sending, upon creation, the multiple translation packages to a human translator to review the multiple reduced properties files.

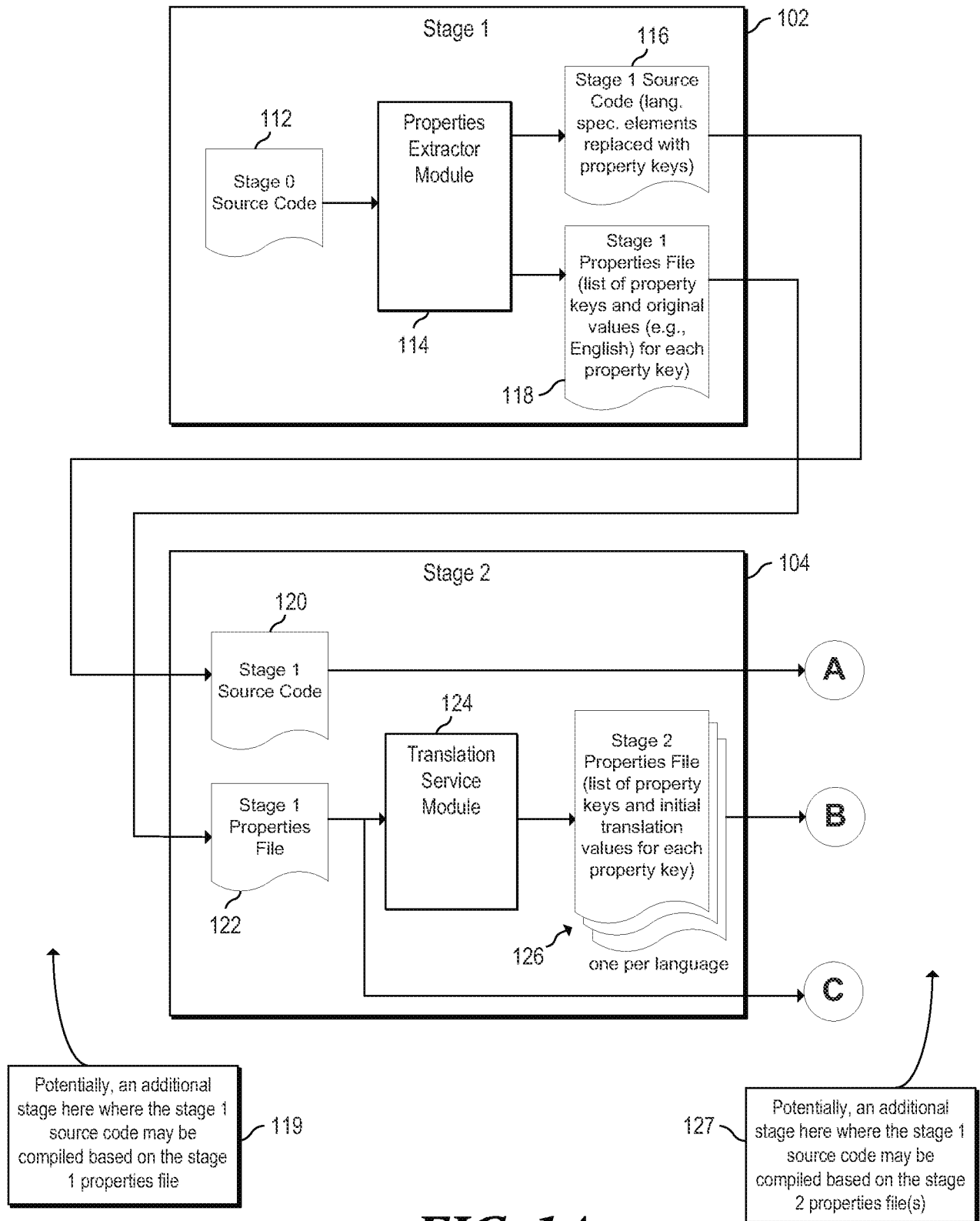


FIG. 1A

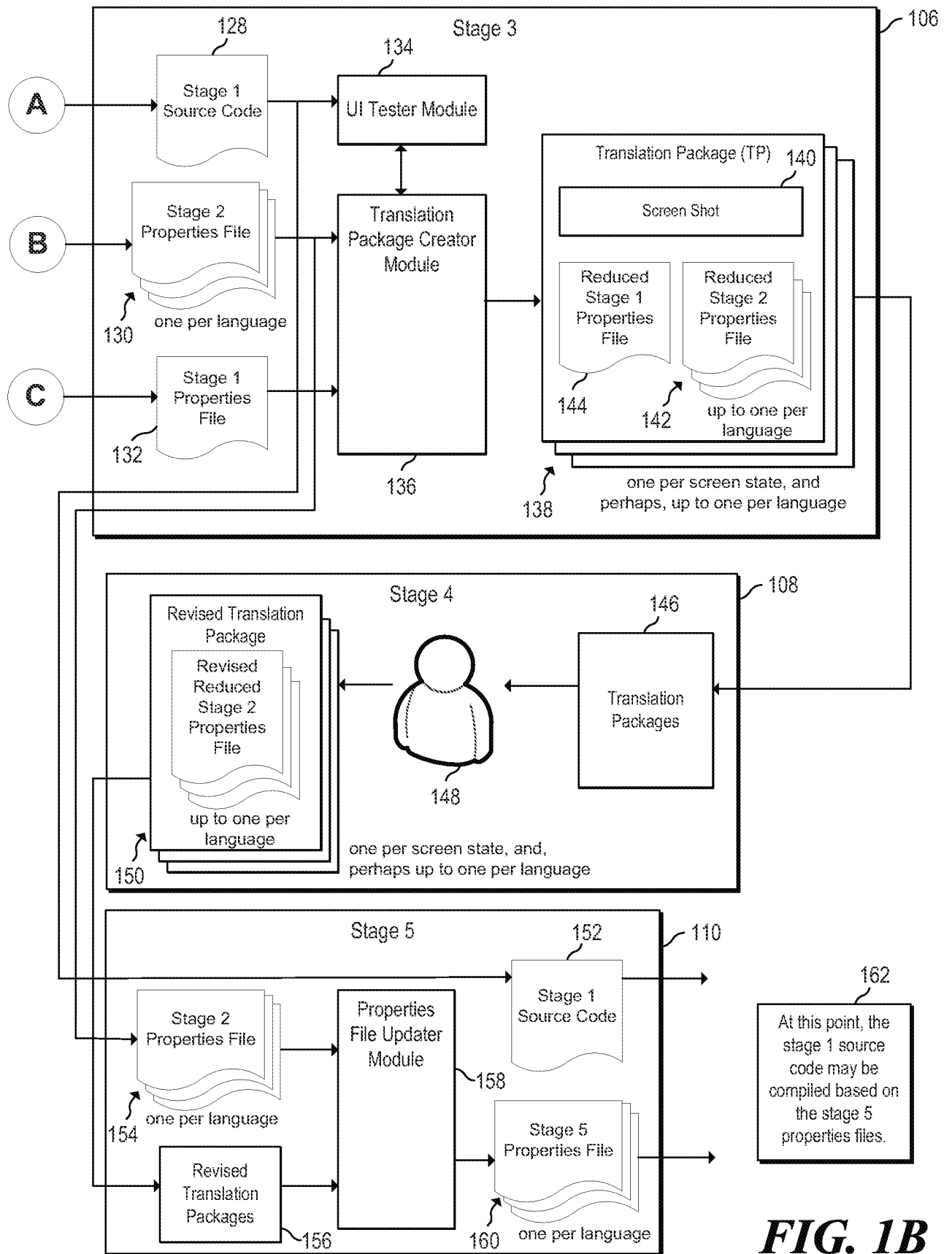


FIG. 1B

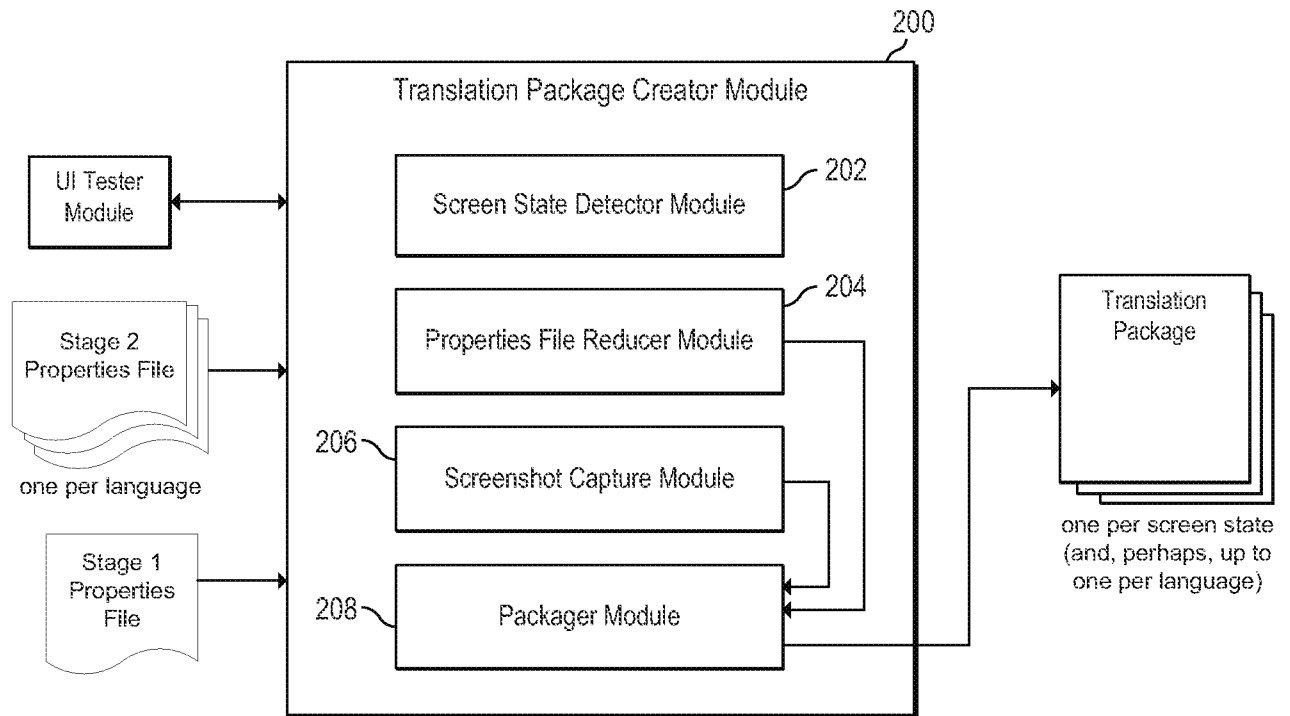


FIG. 2

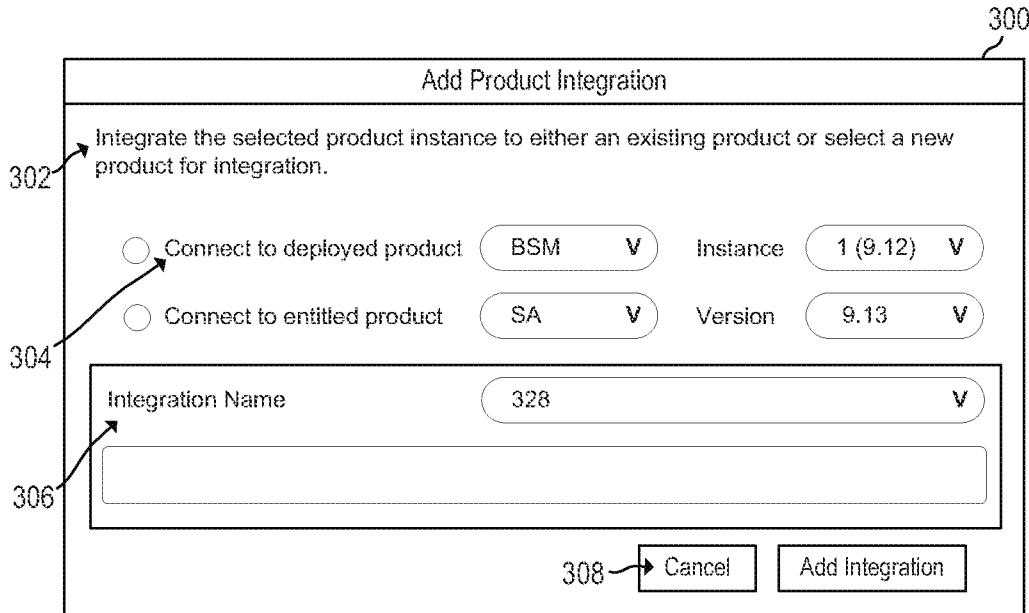


FIG. 3A

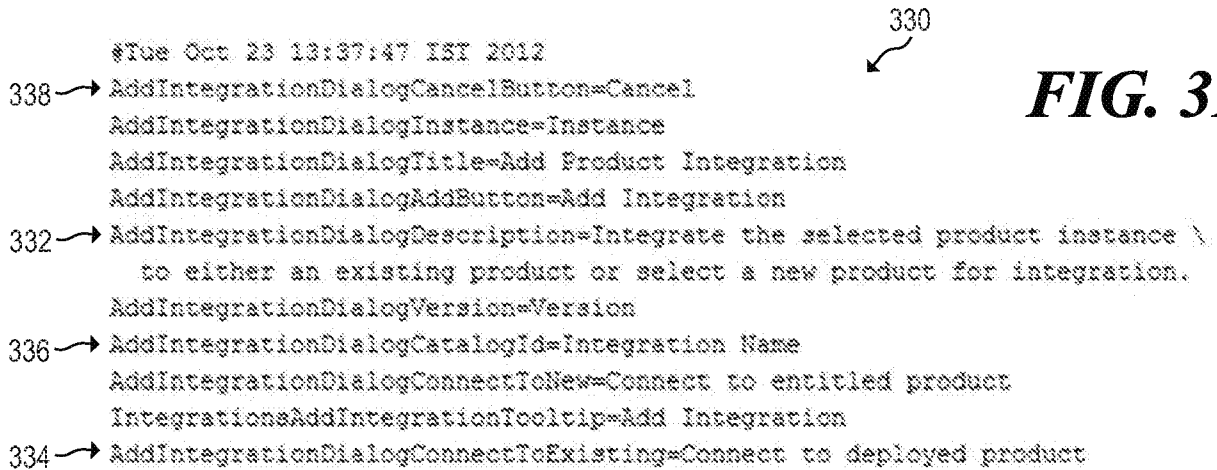


FIG. 3B

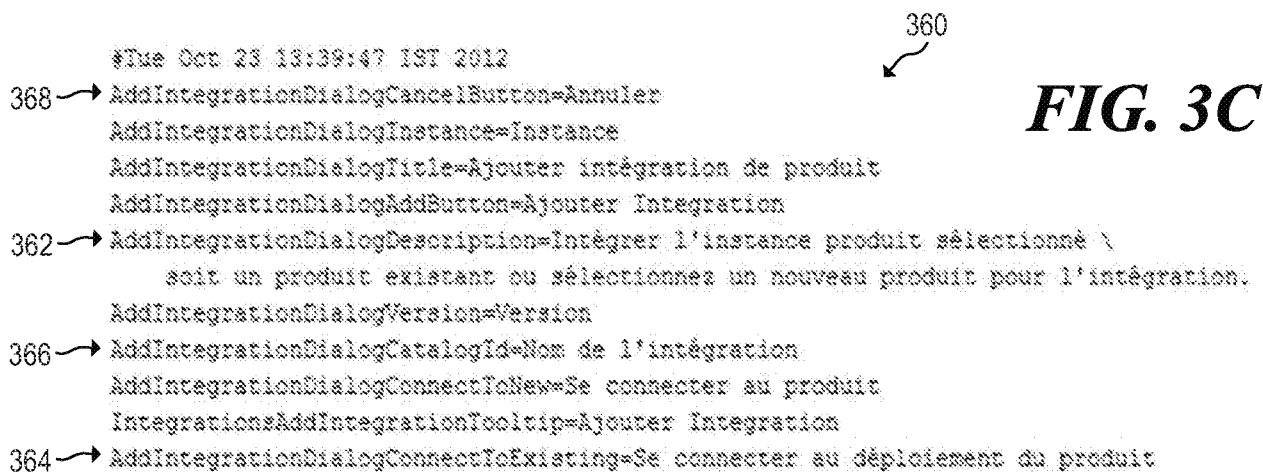
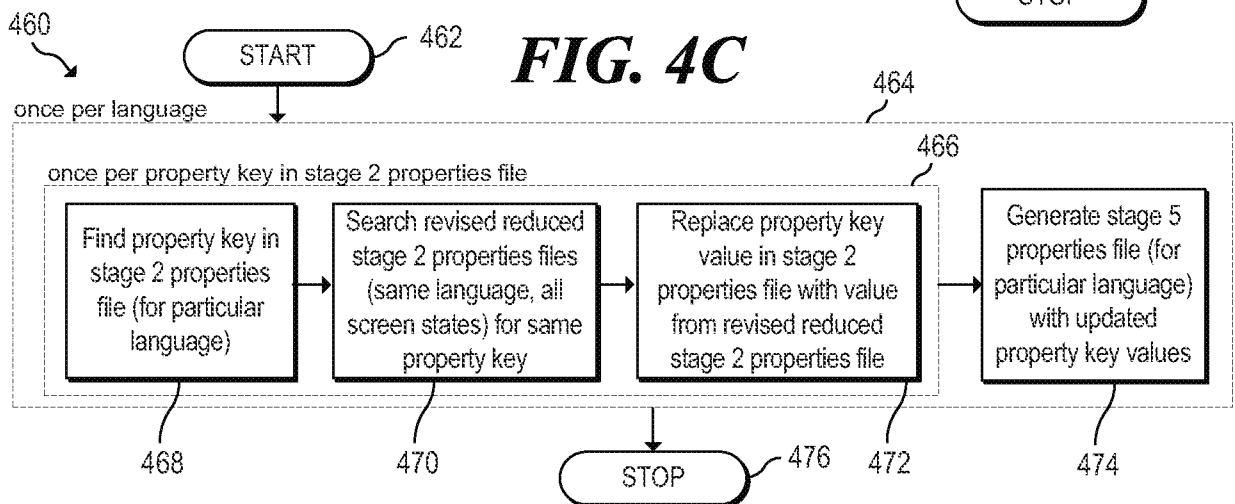
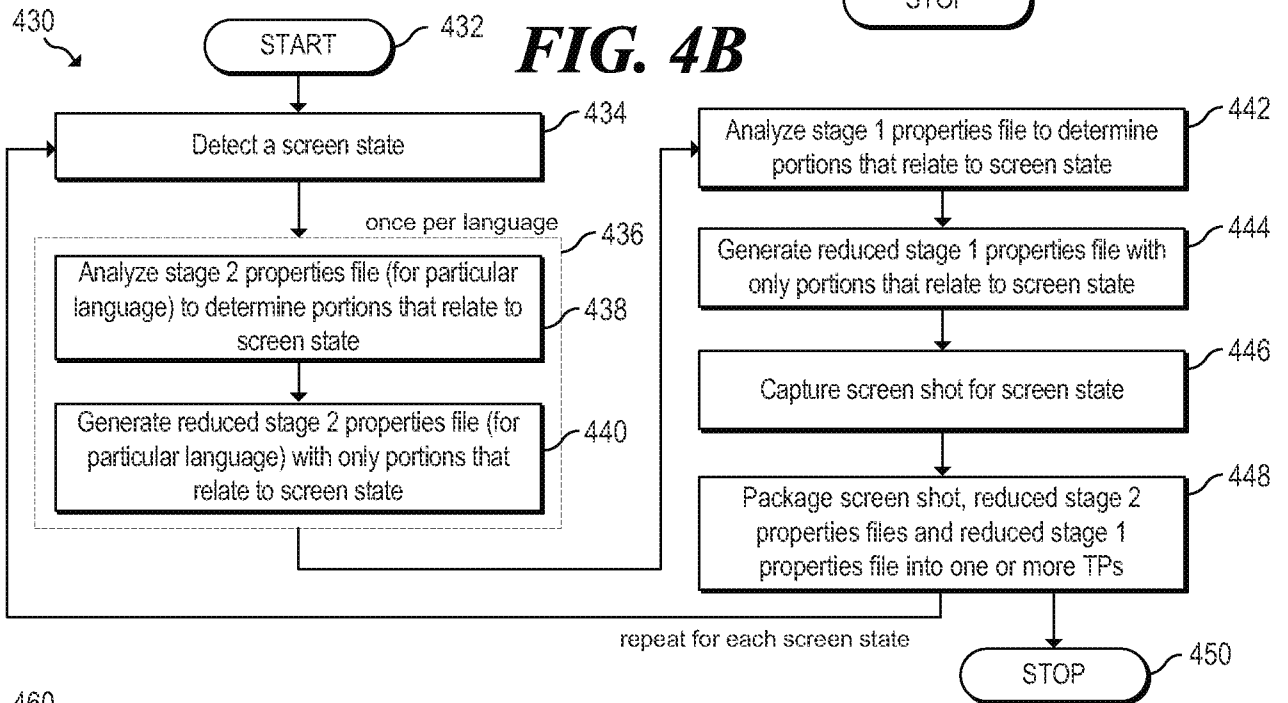
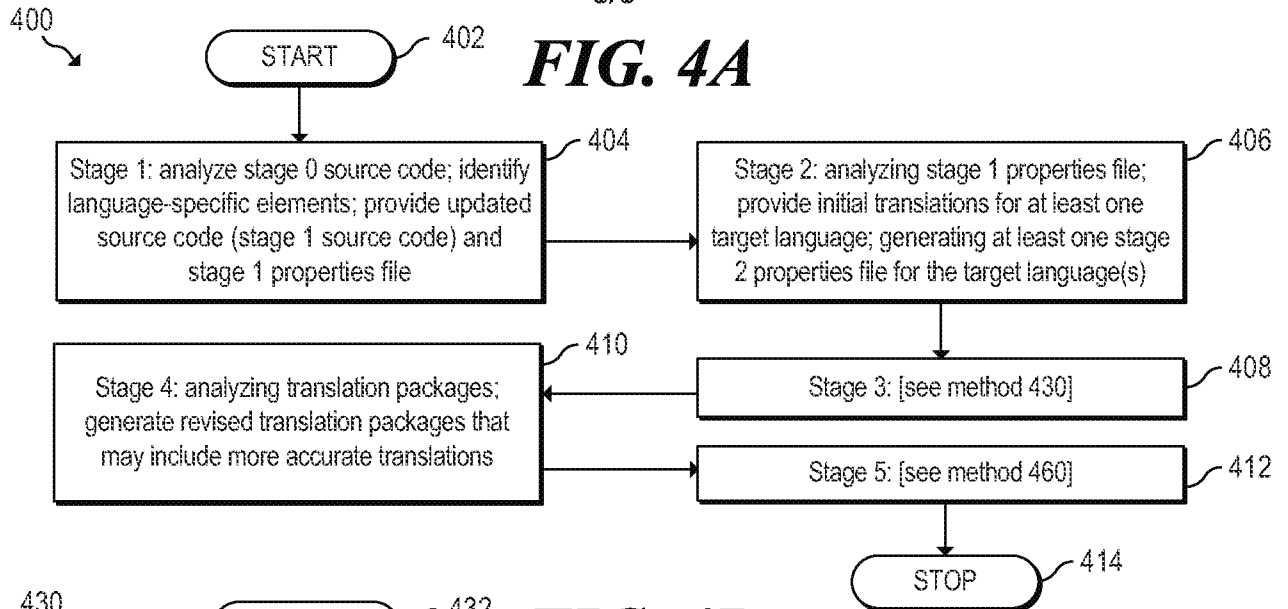


FIG. 3C

5/6



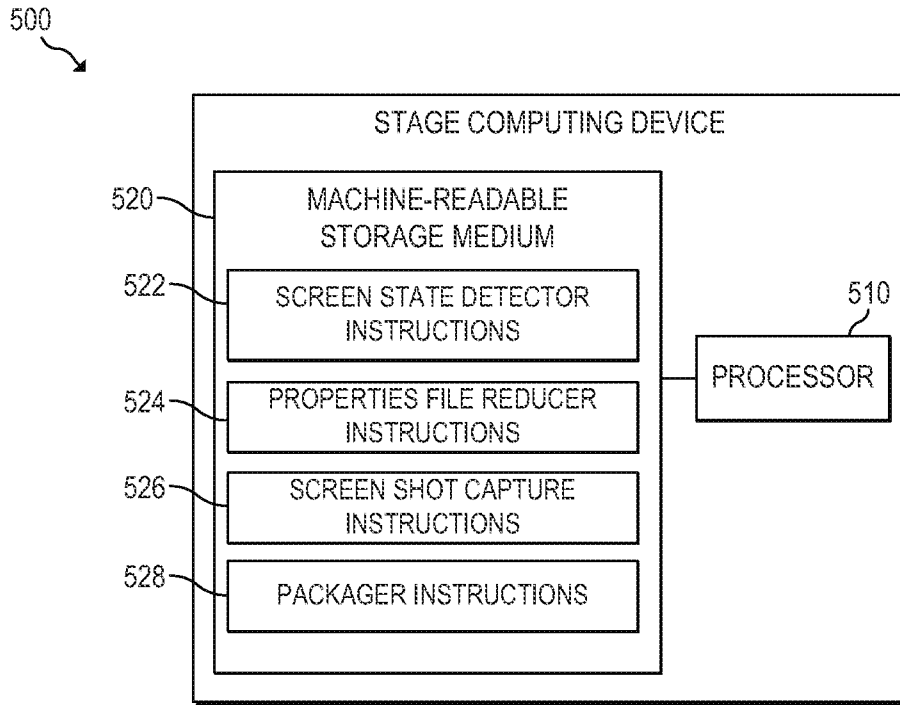


FIG. 5

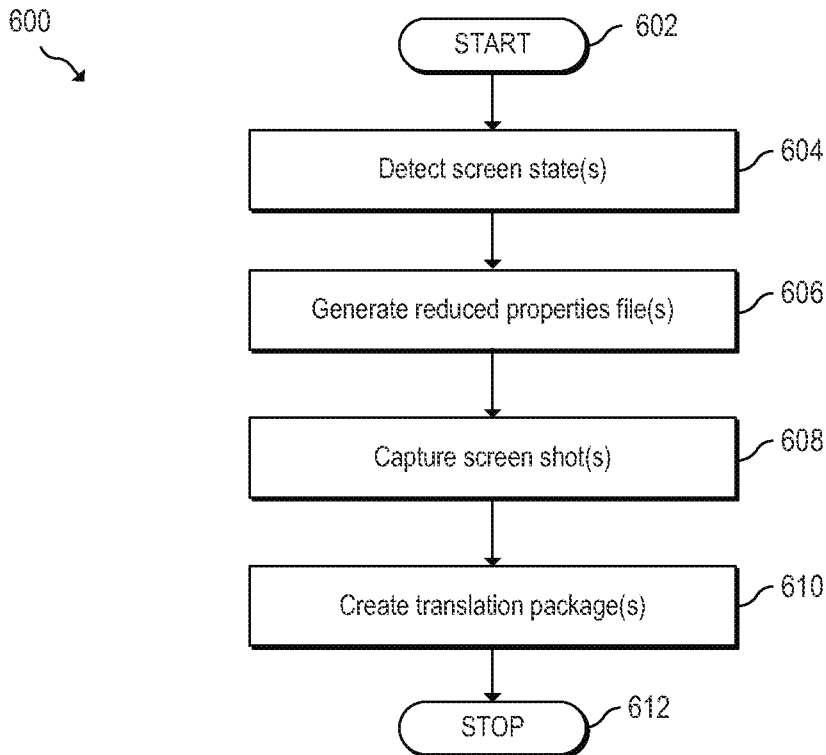


FIG. 6

A. CLASSIFICATION OF SUBJECT MATTER**G06F 17/00(2006.01)i, G06F 17/20(2006.01)i**

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 17/00; G06F 3/048; G06F 17/20; G06F 9/44; G06F 7/00; G06F 17/28

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS(KIPO internal) & Keywords:software, localization, contextual, source, code, language, element, string, capture, screen, property, key, translation, package

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP 1315086 A1 (SUN MICROSYSTEMS, INC.) 28 May 2003 See paragraphs [0011]-[0014], [0018]-[0025], [0034], [0041]-[0044], and [0048]-[0060]; and figures 1-5.	1-16
A	US 7,987,087 B2 (DAVID P. RICH) 26 July 2011 See column 1, line 41 - column 2, line 24; column 3, line 7 - column 5, line 34; column 6, line 59 - column 7, line 40; and figures 1-2.	1-16
A	US 2008-0127045 A1 (DAVID PRATT et al.) 29 May 2008 See paragraphs [0002]-[0010], [0020]-[0026], [0030]-[0034], and [0044]-[0049]; and figures 1-3.	1-16
A	US 2009-0276206 A1 (COLIN FITZPATRICK et al.) 05 November 2009 See paragraphs [0002]-[0004], [0006]-[0007], [0021]-[0023], [0049]-[0052], and [0053]-[0055]; and figures 2-4.	1-16
A	WO 2001-008006 A1 (INTERNATIONAL BUSINESS MACHINES CORPORATION) 01 February 2001 See page 1, line 20 - page 3, line 5; page 4, line 10 - page 5, line 35; and figure 1.	1-16

 Further documents are listed in the continuation of Box C. See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

29 January 2014 (29.01.2014)

Date of mailing of the international search report

03 February 2014 (03.02.2014)

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City,
302-701, Republic of Korea

Facsimile No. +82-42-472-7140

Authorized officer

NHO, Ji Myong

Telephone No. +82-42-481-8528



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2013/036183

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 1315086 A1	28/05/2003	EP 1315084 A1	28/05/2003
		EP 1315085 A1	28/05/2003
		EP 1315085 B1	11/01/2006
		EP 1315086 B1	05/07/2006
		US 2003-0115552 A1	19/06/2003
		US 2003-0126559 A1	03/07/2003
		US 7447624 B2	04/11/2008
US 7987087 B2	26/07/2011	US 2006-287844 A1	21/12/2006
US 2008-0127045 A1	29/05/2008	US 7886267 B2	08/02/2011
US 2009-0276206 A1	05/11/2009	CN 101473318 A	01/07/2009
		EP 1870804 A1	26/12/2007
		KR 10-2009-0020621 A	26/02/2009
		WO 2007-149152 A1	27/12/2007
WO 01-08006 A1	01/02/2001	AU 2000-61677 A1	13/02/2001
		CN 1364260 A	14/08/2002
		EP 1204919 A1	15/05/2002
		IL 147630 A	24/07/2007
		JP 2003-505787 A	12/02/2003
		KR 10-0498529 B1	01/07/2005
		TW 494342 A	11/07/2002
		US 6275790 B1	14/08/2001